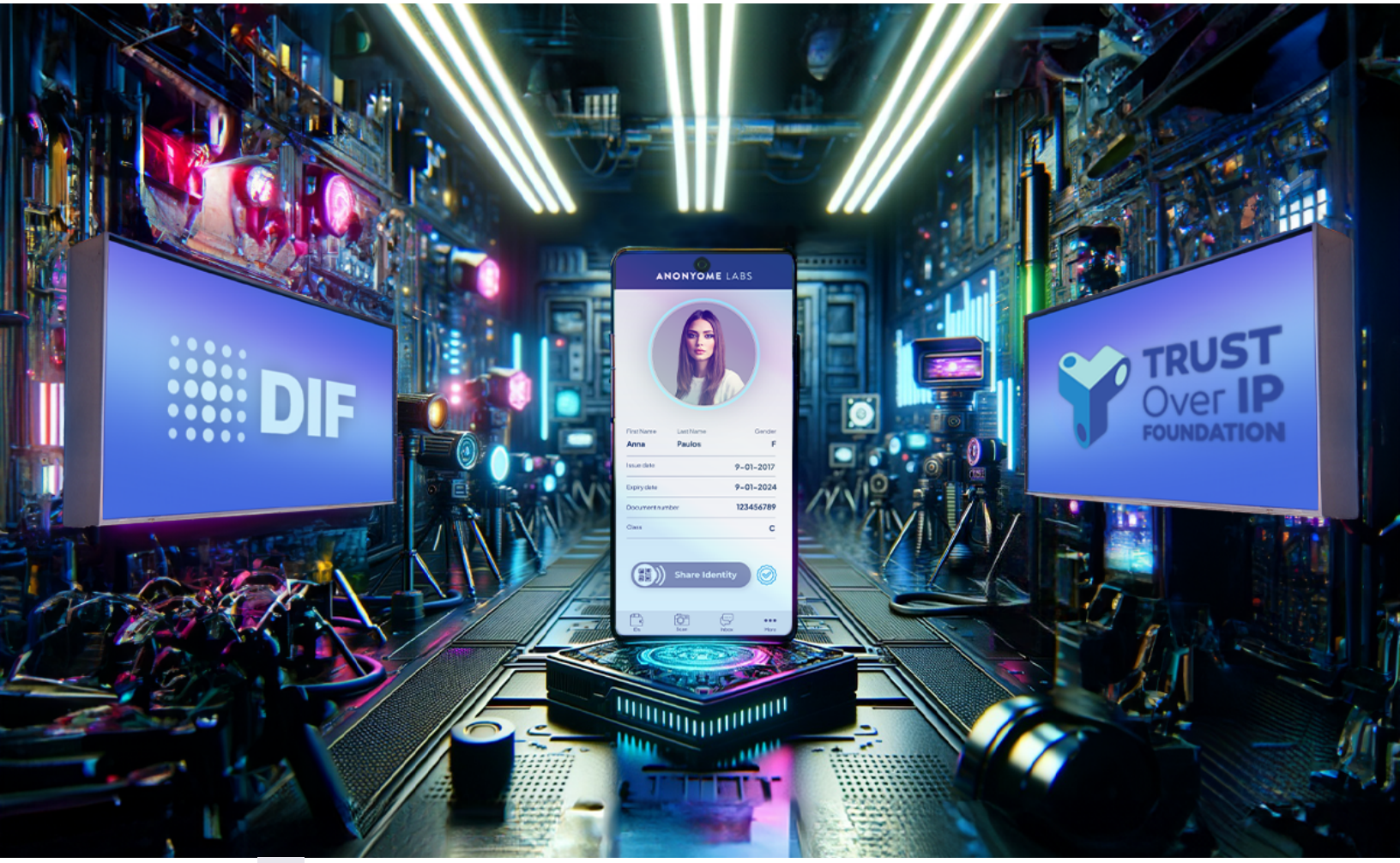
Comparing Two of Decentralized Identity's Leading Governance Models

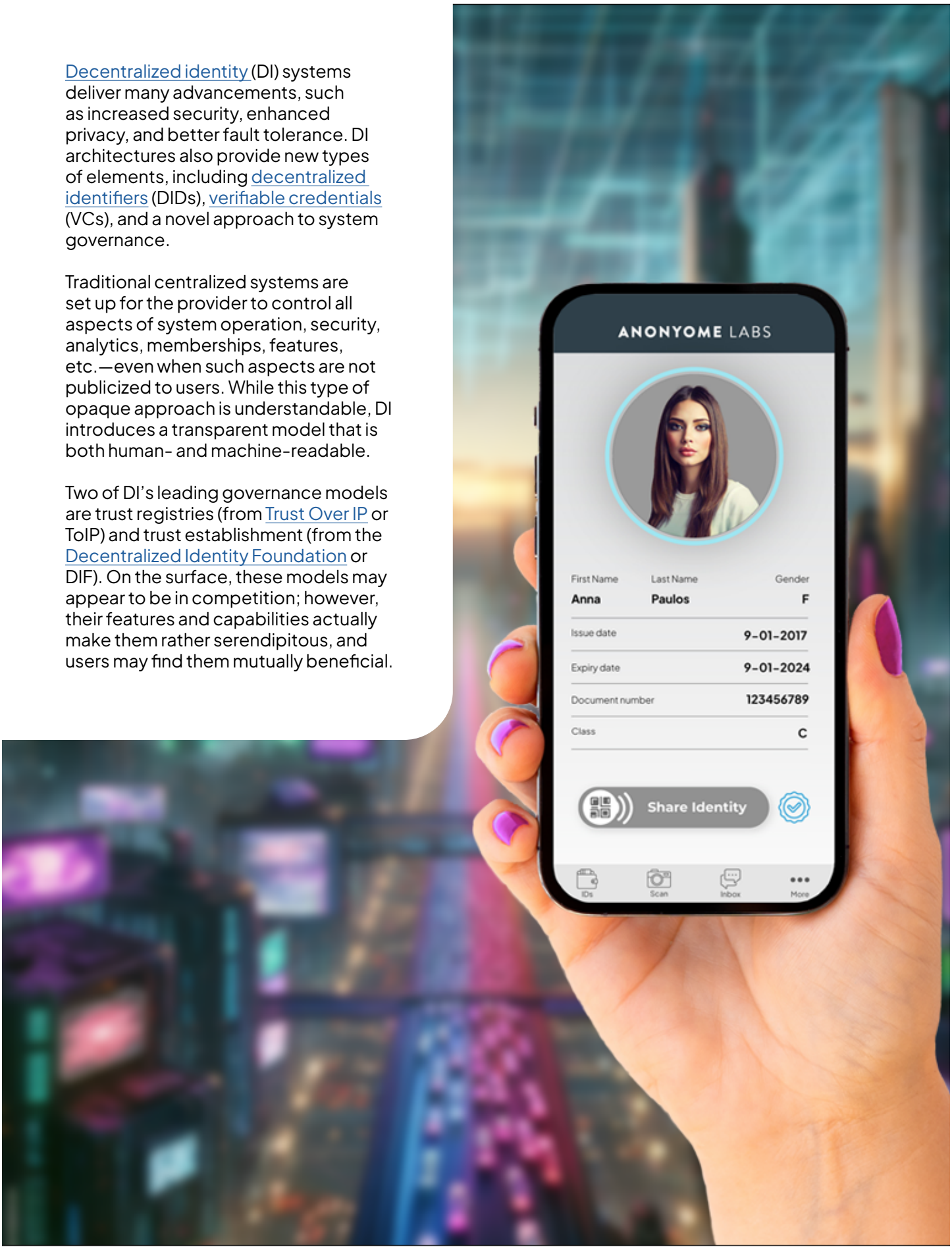


ANONYOME LABS

Decentralized identity (DI) systems deliver many advancements, such as increased security, enhanced privacy, and better fault tolerance. DI architectures also provide new types of elements, including decentralized identifiers (DIDs), verifiable credentials (VCs), and a novel approach to system governance.

Traditional centralized systems are set up for the provider to control all aspects of system operation, security, analytics, memberships, features, etc.—even when such aspects are not publicized to users. While this type of opaque approach is understandable, DI introduces a transparent model that is both human- and machine-readable.

Two of DI's leading governance models are trust registries (from Trust Over IP or ToIP) and trust establishment (from the Decentralized Identity Foundation or DIF). On the surface, these models may appear to be in competition; however, their features and capabilities actually make them rather serendipitous, and users may find them mutually beneficial.



What is a trust registry?

ToIP's Governance Stack Working Group created the Trust Registry model, which presents the governance concepts using the four-layer ToIP architecture stack:

In the ToIP model, the four layers define a full technology architecture ecosystem:

Layer 1: defines a class of interoperable public utilities that enable DIDs to be immutably correlated with author-provided data (e.g., cryptographic keys, communication addresses, etc.). Layer 1 utilities usually take the form of a cryptographic ledger or blockchain.

Layer 2: defines one or more methods by which ecosystem participants may communicate securely. Arguably the most advanced Layer 2 communication method is DIDComm from DIF.

Layer 3: introduces VCs along with methods of issuing, holding, and verifying them. Leading VC standards include W3C Verifiable Credentials and AnonCreds from Hyperledger.

Layer 4: describes application ecosystems, which are loosely analogous to large enterprise platforms or multiple interoperable platforms.

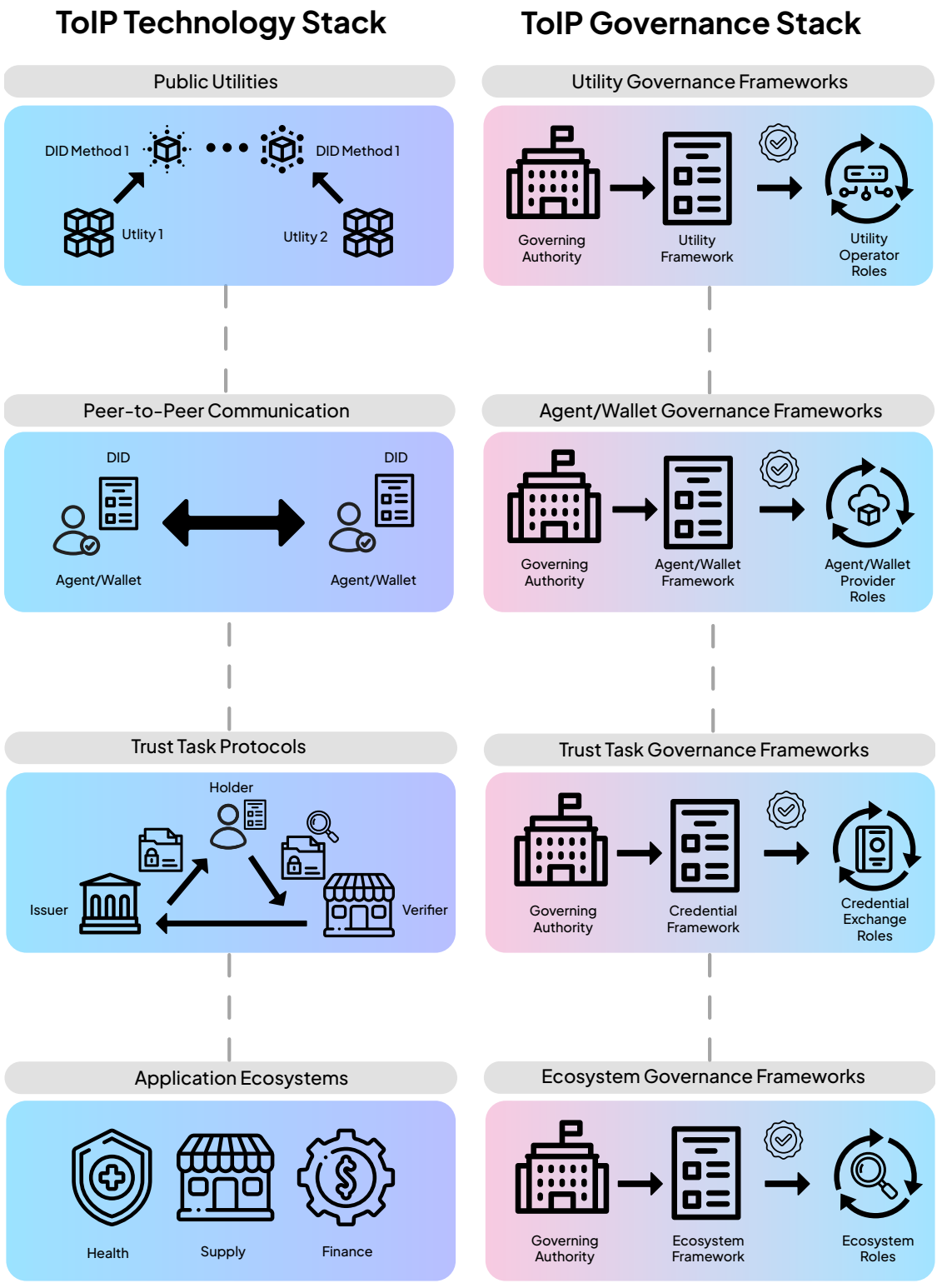


Figure 1 – ToIP architecture stack

While the ToIP architectural model will be described in other sources, this high-level description has been given to assist in understanding how governance is applied to this stack. As depicted on the right side of the ToIP architecture stack (see Figure 1), a separate governance model is uniquely applied to each layer. For example, a Layer 1 ledger will have its own governance model that is separate from a Layer 2 communication protocol that uses Layer 1 services. Similarly, Layer 3 and 4 services will also have their own unique governance models. Layer 4 also introduces a trust registry, which will be described below. Since the governance models of upper layers may need to anticipate certain operational aspects of underlying layers, they may contain some governance requirements that describe their methods of interaction or reliance on the lower layers.

ToIP's approach is to create a governance framework (GF) that guides organizations in creating their own governance model more than specifying exactly what rules and descriptions a governance model must contain. In other words, it is a process for creating a governance model rather than a pre-existing governance model to be applied. This process may even evolve into a certification process that evaluates how well an organization's governance model conforms to recommended industry standards. There are several steps in this process, which are encapsulated in the ToIP Governance Metamodel Specification. The metamodel describes the process for creating:

1. **Primary document** – This serves as the "home page" for the GF that specifies an identifying DID, a description of the system being governed, which ToIP document versions are being used, links to external sites, governing authority descriptions, administering authority, purpose, scope, objectives, principles, requirements, revisions, extensions, and a list of controlled documents.

2. **Controlled documents** (each document is optional) are –
    a. **Glossary**
    b. **Risk assessment**
    c. **Trust assurance and certification**
    d. **Governance requirements**
    e. **Business requirements**
    f. **Technical requirements**
    g. **Information trust requirements**
    h. **Inclusion, equitability, and accessibility requirements**
    i. **Legal agreements**

This further demonstrates that what ToIP has created isn't a governance model per se, but rather the process for creating one. Quite often, teams creating DI systems don't know where to start when defining governance for their systems and the ToIP model is an excellent roadmap. This approach should be very familiar to software engineers who have used similar document-oriented roadmap definition processes when developing large software systems. According to the ToIP architecture stack diagram (see Figure 1), this process can be performed at each of the four ToIP layers.

Once a governing authority has developed a ToIP GF, it will make it available to ecosystem participants. Figure 2 gives a high-level view of the ToIP operational ecosystem and highlights the governing authority's trust registry (i.e., as defined by Layer 4 in the ToIP governance stack):

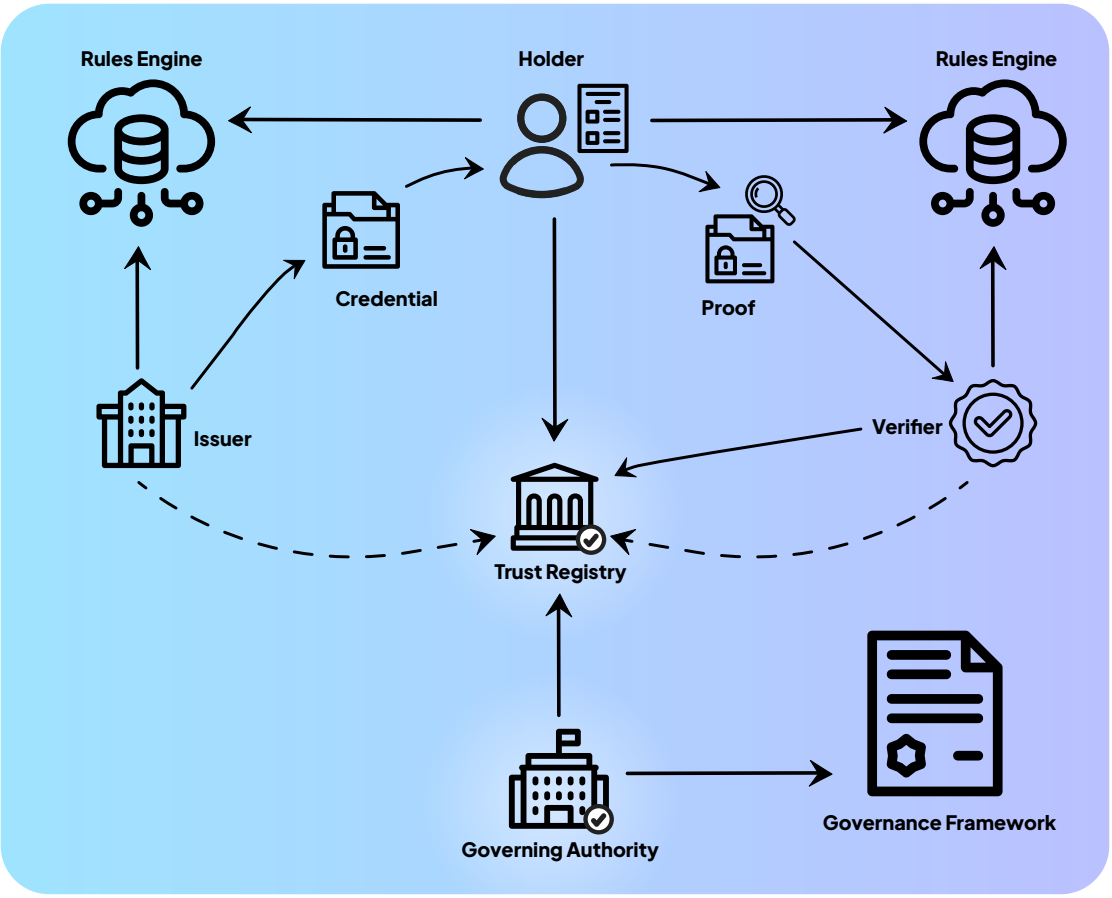


Figure 2 – ToIP operational ecosystem overview

As shown in Figure 2, the trust registry is embodied as an internet-accessible service endpoint (e.g., server) that provides a callable API for participant processes to invoke during the issuance and usage of VCs. This is described in the Trust Registry Protocol V1 Specification as:

"A core role within Layer 4 of the ToIP stack is a trust registry … This is a network service that enables the governing authority for an ecosystem governance framework (EGF) to specify what governed parties are authorized to perform what actions under the EGF."

Using this process, issuers and verifiers will register with the trust registry. When a verifier requests a credential proof from a holder, the holder can query the trust registry to ensure the integrity of the VC proof request process. Similarly, when a verifier receives a VC proof, it can query the trust registry to ensure the integrity of the proof expression. Using this operational model, the trust registry performs a pivotal role in the VC issuance and usage processes.

What is trust establishment?

While ToIP's GF processes appear best suited for enterprise-level ecosystem efforts, the trust establishment (TE) processes that DIF is creating are intended to be much simpler. According to the Trust Establishment 1.0 document:

"Supporting trust decisions in decentralized identity architectures, particularly open-world architectures, is a problem that many have tried to solve. This specification aims to describe a practical, interoperable building block for supporting multiple different kinds of trust-decision Trust Establishment solutions. We define here a lightweight trust document: a means by which a Party communicates their assertions for a Topic about a set of Parties."

The TE document further describes:
"This specification describes only the data model of trust documents and is not opinionated on document integrity, format, publication, or discovery."

Rather than presenting a series of processes by which a GF can produce a governance model, the DIF specification provides a single "lightweight trust document" that produces a governance data model.

Since the TE does not require a particular data format, it can be embodied in many formats. In one instance, it can be used through an internet-accessible API as is specified for the ToIP trust registry/governance model solution. However, it is most commonly described as a cryptographically signed and JSON-formatted document that can be downloaded from a website, immutable data source, or a provider's own service.

The TE is a newly emerging specification and will likely undergo many enhancements and updates. At present, Section 5 of the TE describes the data model and provides a series of required and optional properties, as follows:

- **id** – The object MUST contain an id property. The value of this property MUST be a string. The string SHOULD provide a unique ID for the desired context.
- **author** – The object MUST contain an author property. The value of this property MUST be a string value representing the DID of the author.
- **created** – The object MUST contain a created property proving a date-time value for when the object was created. The value of this property MUST be an RFC 3339 compliant timestamp value.
- **validFrom** – The object MUST contain a validFrom property proving a date-time value for when the object is to be used. The value of this property MUST be an RFC 3339 compliant timestamp value.
- **validUntil** – The object MAY contain a validUntil property proving a date-time value for when the object is no longer to be used. The value of this property MUST be an RFC 3339 compliant timestamp value.
- **version** – The object MAY contain a version property. If present, the value of this property MUST be a string that uniquely identifies the instance of this document.
- **entries** – The object MUST contain an entries property that represents combinations of topics and entities for trust statements. Its value MUST be a JSON object composed as follows:
  - The object MUST have map keys as string values identifying the topic of the TE document.
  - The object MUST have map values as JSON objects, containing JSON maps and MUST be composed as follows:
    - MUST have map keys as DIDs which identify parties for which trust is being expressed.
    - MUST have map values as JSON objects conforming to the associated schema of the parent topic value.

Using JSON formatting, Figure 3 provides an example of how several of the above properties would appear:

```
{
 “id”: “32f54163-7166-48f1-93d8-ff217bdb0653”,
 “author”: “did:example:alice”,
 “created”: “2022-04-20T04:20:00Z”,
 “version”: “0.0.3”,
 “entries”: {
  “https://example.com/trusted-supplier.schema.json”: {
   “did:example:bob”: {
    “on_time_percentage”: 92,
    “goods”: [“basmati”, “jasmine”, “sushi”]
   },
   “did:example:carol”: {
    “on_time_percentage”: 74,
    “goods”: [“long-grain”, “short-grain”, “extra glutinous”]
   }
  },
  “https://example.com/other.schema.json”:{
   “did:example:bob”: {
    “foo”: “bar”
   },
   “did:example:carol”: {
    “foo”: “baz”
   }
  }
 }
}
```

Figure 3 – TE properties in JSON format

Building upon the previous depiction, Figure 4 illustrates a more complex trust document that in this instance is packaged as a verifiable credential:


```
{
 “@context”: [
  “https://www.w3.org/2018/credentials/v1”,
  “https://www.w3.org/2018/credentials/examples/v1”
 ],
 “id”: “http://example.edu/credentials/3732”,
 “type”: [“VerifiableCredential”, “TrustEstablishment”, “TrustedSuppliers”],
 “issuer”: “did:example:alice”,
 “issuanceDate”: “2010-01-01T00:00:00Z”,
 “credentialSubject”: {
  “id”: “did:example:ebfeb1f712ebc6f1c276e12ec21”,
  “trustEstablishment”: {
   “id”: “32f54163-7166-48f1-93d8-ff217bdb0653”,
   “author”: “did:example:alice”,
   “created”: “2010-01-01T19:23:24Z”,
   “version”: “0.0.3”,
   “entries”: {
    “https://example.com/trusted-supplier.schema.json”: {
     “did:example:bob”: {
      “on_time_percentage”: 92,
      “goods”: [“basmati”, “jasmine”, “sushi”]
     },
     “did:example:carol”: {
      “on_time_percentage”: 74,
      “goods”: [“short-grain”, “long-grain”, “extra glutinous”]
     }
    }
   }
  }
 },
 “proof”: {
          “type”: “Ed25519Signature2020”,
  “created”: “2021-11-13T18:19:39Z”,
  “verificationMethod”: “did:example:alice#key-1”,
  “proofPurpose”: “assertionMethod”,
  “proofValue”:
“z58DAdFfa9SkqZMVPxAQpic7ndSayn1PzZs6ZjWp1CktyGesjuTSwRdoWhAfGFCF5bppETSTojQCrfFPP2oumHKtz”
 }
}
```


Figure 4 – Trust document as a verifiable credential proofed by data integrity

Using the trust document specification as described enables the rapid creation of governance/trust models that can be quickly used in a wide variety of platforms.

Since the TE document specification is only intended to describe creating a data model, how it is published or delivered is left up to the implementer's discretion. That said, the specification gives a couple of considerations for delivery:

1. **Known locations** – These enable TE documents to be located via crawling a known set of endpoints or some other common routing mechanisms; DIDs with service endpoints offer one abstraction for layering these and integrating them into discovery processes.
2. **Query target entities** – These enable querying for all or some subset of TE documents from a target entity via common DID-based data query/interaction protocols.

If deployed to known locations (e.g., a provider's GitHub page, website, etc.), calling entities need to know where to look (e.g., DID or URI) and the protocol for requesting the document. If deployed on the target entity itself, the TE document is easily referenced in the DID document of the target entity's DID. In either case, the DID used to sign the TE (i.e., create its digital signature) must be anchored in a secure location, such as an immutable ledger. This will provide reliable security integrity for the TE while enabling it to be stored in a conveniently accessible location rather than on an immutable data source itself.

Regardless of the implementation, the TE specification's designers intended for its preferred delivery method to be a complete download of the document as a whole. With the downloaded document, a requesting entity can archive it for continued use and only request updates periodically.

How do the ToIP and DIF solutions compare?

The ToIP solution is architected to align with the software engineering processes of larger entities while the DIF solution is intended to fulfill targeted development needs. As such, they are difficult to compare since they do not have the same elements and address different organizational requirements.

Despite their difference in target applications, there is some potential serendipity. The ToIP solution emphasizes an engineering approach for creating the eventual governance data model. Despite its simplicity, the DIF approach gets to the heart of creating the data model given the goals of the model's designer. Given that the ToIP processes emphasize how to design the data model while the DIF solution emphasizes how to embody the data model once it's created, perhaps they will follow each other through the development process. From this perspective, the ToIP processes could define what needs to go into a governance data model while the DIF solution could define how that is to be done. This perspective moves the respective solutions from alternatives to parts of the same process.

Regardless of whether the respective paradigms are used jointly or separately, there is one caution that must be pointed out: The web-accessible API model that trust registries employ enables targeted requests for data controlled under the governance model. This may create a phone home scenario where requestors are providing subject, sequence, or timing information to the trust registry and this could be used to track individual usage and cause a number of privacy issues. Conversely, the single download option that the TE specification presents enables requestors to download the TE once and reuse it many times. This difference eliminates the phone home upon verification scenario, which has been the subject of recent privacy concerns. To aid in privacy protection, options such as download once, use many, regularly scheduled TE retrieval, or other connectionless verifications are recommended.

With that caution noted, both of these governance creation and usage scenarios provide tremendous improvement over the centralized governance models that exist today. If used together, the ToIP solution and the DIF model appear mutually beneficial and provide different benefits to the decentralized governance operating goals.